M 2015

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

MOBILE APPLICATION FOR BLOCKING SPAM CALLERS

RUI MIGUEL TORRES DA COSTA RODRIGUES CARDOSO

DISSERTAÇÃO DE MESTRADO APRESENTADA
À FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO EM
ÁREA CIENTÍFICA

U.PORTO
FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

MIEEC - MESTRADO INTEGRADO EM ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES | **2014/2015**

A Dissertação intitulada

"Mobile Application for Blocking Spam Callers"

foi aprovada em provas realizadas em 18-02-2015

o júri

Presidente Professor Doutor Manuel Alberto Pereira Ricardo
Professor Associado do Departamento de Engenharia Eletrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

Professor Doutor Jorge Botelho da Costa Mamede
Professor Adjunto Departamento de Engenharia Eletrotécnica do Instituto Superior de Engenharia do Porto

Professor Doutor Ricardo Santos Morla
Professor Auxiliar do Departamento de Engenharia Eletrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

O autor declara que a presente dissertação (ou relatório de projeto) é da sua exclusiva autoria e foi escrita sem qualquer apoio externo não explicitamente autorizado. Os resultados, ideias, parágrafos, ou outros extratos tomados de ou inspirados em trabalhos de outros autores, e demais referências bibliográficas usadas, são corretamente citados.

Autor - Rui Miguel Torres da Costa Rodrigues Cardoso

Faculdade de Engenharia da Universidade do Porto

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

FEUP

Mobile Application for Blocking Spam Callers

Rui Miguel Torres da Costa Rodrigues Cardoso

Mestrado Integrado em Engenharia Electrotécnica e de Computadores

Supervisor: Ricardo S. Morla

Co-Supervisor: Muhammad A. Azad

26 January 2015

Abstract

VoIP (Voice over Internet Protocol) technology offers cheap telephony service and its openness can lead it to denial of service and service abuse attacks from the malicious users. One such example of such attack is the Spam over Internet Telephony (SPIT), which can be done by bulk calls and messages sent by Telemarketers, prank callers, and advertisers. Some challenges emerge when developing tools for SPIT detection. For example: the contents of a voice call are only disclosed when it is answered; the time it takes to decide if the caller is spammer or not spammer must be the small. The solution proposed by this dissertation to detect spam calls in a mobile phone (designated as App from now onwards) uses the Caller-Rep algorithm. Caller-REP calculates the reputation of the caller using Caller's direct trust with their called people. The direct trust between caller and callee is computed using a number of features that includes call-rate, call duration in both directions and out-degree of the caller. The global reputation of the caller is then computed using Eigen trust algorithm. The reputation of the caller is then used to distinguish spammers from non-spammers. In this dissertation we focused on how mobile user collaborates with centralized server to achieve the motives of collaborative spam detection. To this extent, a suitable protocol was developed for the connection between the server and the mobile phone that is being used to update server with trust values of each mobile users with the user he receives and made calls. The centralized server aggregates the direct trust scores from participating users, computes the global reputation of each user and decides the caller status by thresholding global reputation scores. The *App* is user friendly and provides many value added features to the users. The whole work took into account a cooperative information-sharing between users and the server providing each of them with information (list of Direct Trust) for the construction of the blacklist.

Resumo

A tecnologia VoIP (Voice over Internet Protocol) oferece um serviço de telefonia de baixo custo e, o facto de ser *freeware*, pode conduzir a estados de DoS (Denial of Service) bem como a ataques por parte de utilizadores mal-intencionados. Um exemplo desses ataques é o Spam over Internet Telephony (SPIT), podendo ser causado através de chamadas e mensagens em massa realizadas por empresas de publicidade e telemarketing e hackers. Deste modo, alguns desafios surgem aquando do desenvolvimento de ferramentas para detecção de SPIT. Por exemplo: o conteúdo de uma chamada de voz é apenas divulgado quando esta é atendida; o tempo que leva para decidir se um número é ou não spammer deve ser o mais curto possível. A solução proposta por esta dissertação, para detectar chamadas de spam num telefone móvel, (designado como *App* a partir de agora) usa o algoritmo Caller-Rep. Este algoritmo calcula a reputação do autor da chamada usando a confiança direta que o mesmo tem com os seus contactos. A confiança direta entre o chamador e o receptor é calculado usando uma série de parâmetros que incluem a taxa de chamadas, a duração das chamadas em ambas as direções e o *out-degree* do chamador. A reputação global do chamador é então calculada usando o algoritmo de confiança Eigen. A reputação do autor da chamada é então usado para distinguir os utilizadores spammers dos não-spammers. Nesta dissertação, é dado especial interesse ao modo como o utilizador móvel colabora com o servidor centralizado para dar origem a um sistema de detecção de chamadas de voz spam colaborativo. Neste sentido, foi desenvolvido um protocolo adequado para a ligação entre o servidor e o telefone móvel para permitir a atualização do servidor com valores de confiança de cada utilizador móvel. O servidor centralizado agrega os valores de confiança direta dos utilizadores registados no sistema, calcula a reputação global de cada utilziador e decide que utilizador é ou não spammer com base na sua reputação. A *App* é de fácil utilização e oferece muitos recursos de valor acrescentado aos seus utilizadores. Todo o trabalho realizado teve em linha de conta uma perspetiva de partilha de informação entre os diferentes utilizadores e o servidor, contibuindo, cada um deles, com informação para uma construção colaborativa da lista de spammers.

Agradecimentos

Ao meu professor e orientador Ricardo Morla pelas indicações dadas ao longo de todo o trabalho desenvolvido e disponibilidade demonstrada na resolução de problemas que foram surgindo.

Ao Ajmal que sempre se demonstrou disponível para me atender e debater ideias, bem como propor algumas soluções para alcançar certos objetivos.

A todos os meus amigos que sempre me ajudaram quando necessário.

Aos meus pais, irmãos, sogros e cunhados e toda a minha família que demonstraram sempre um apoio incondicional ao longo do meu percurso académico e, em especial, ao longo da realização desta dissertação.

À Adalgisa que sempre me apoiou e deu força ao longo deste trabalho e de todo o meu percurso académico.

Rui Cardoso

Contents

Agradecimentos v

1 Introduction 1
- 1.1 Motivation . . . 1
- 1.2 Problem . . . 1
- 1.3 Contribution . . . 2

2 Technical Background and Related Work 5
- 2.1 Technical Background . . . 5
  - 2.1.1 VoIP . . . 5
  - 2.1.2 Advantages of VoIP . . . 7
  - 2.1.3 Network concerns for VoIP services . . . 7
- 2.2 VoIP Security Threats . . . 8
- 2.3 Spam over Internet Telephony (SPIT) . . . 8
  - 2.3.1 SPIT Threats and Scenarios . . . 9
- 2.4 Android . . . 10
  - 2.4.1 Brief History and Evolution . . . 10
  - 2.4.2 Market Share . . . 12
- 2.5 Anti-Spit Overview . . . 12
  - 2.5.1 Content-based approaches . . . 13
  - 2.5.2 Access list-based approaches . . . 13
  - 2.5.3 Challenge response-based approaches . . . 13
  - 2.5.4 Imposing additional costs on callers . . . 14
  - 2.5.5 Extended call-setup based approaches . . . 14
  - 2.5.6 Social reputation-based approaches . . . 14
- 2.6 Stand-alone System Spam Detection . . . 15
- 2.7 Social network features for VoIP users . . . 16
  - 2.7.1 In-degree and out-degree . . . 16
  - 2.7.2 Call rate . . . 17
  - 2.7.3 Call duration . . . 17
- 2.8 Caller-REP . . . 17
  - 2.8.1 Why Caller-REP? . . . 17
  - 2.8.2 Caller direct trust . . . 19
  - 2.8.3 Caller global reputation . . . 19
  - 2.8.4 SPIT caller detection . . . 20

3 Spam Blocker Architecture **23**
3.1 Application Elements . . . 23
3.1.1 System Architecture . . . 23
3.1.2 Call flow . . . 24
3.1.3 Server Connection . . . 25
3.2 Constraints of the Android OS API . . . 25

4 Implementation and Evaluation **27**
4.1 System Class Diagrams . . . 27
4.1.1 Mobile Application Class Diagram . . . 27
4.1.2 Server Class Diagram . . . 32
4.2 System Databases . . . 35
4.2.1 Mobile Database . . . 35
4.2.2 Server Database . . . 35
4.3 Use Cases . . . 36
4.3.1 Show Contacts . . . 36
4.3.2 Show Blacklist . . . 36
4.3.3 Show History . . . 37
4.3.4 Choose Settings . . . 38
4.3.5 Notification . . . 39
4.4 Performance Results . . . 41

5 Conclusions and Framework Proposal **43**
5.1 Accomplished goals . . . 43
5.2 Framework proposal . . . 44
5.2.1 Direct Trust Module . . . 44
5.2.2 Server Module . . . 44
5.2.3 Share Module . . . 44

A Appendix **45**
A.1 Laptop Specifications . . . 45
A.2 Mobile Phone Specifications . . . 45

List of Figures

1.1 *Why using Mobile Application for Blocking Spam Caller?* 2

2.1 *Smartphone OS Market Share* 11
2.2 *Caller-REP block diagram* 18

3.1 *System Architecture* 23

4.1 *Mobile Application Class Diagram* 28
4.2 *CallLogsProcessing Class* 28
4.3 *DatabaseHandler Class* 29
4.4 *BlacklistDetails Class* 30
4.5 *AddNumberToBlacklistDialog class* 30
4.6 *ContactDetails class* 31
4.7 *CellCallsService class* 32
4.8 *SSLClient class* 32
4.9 *Server side class diagram* 33
4.10 *Server Graphical Interface* 34
4.11 *Relational Model of the Mobile Application Database* 35
4.12 *Relational Model of the Server Database* 35
4.13 *Main screen* 36
4.14 *Contacts screen* 36
4.15 *Main screen* 37
4.16 *Blacklist screen* 37
4.17 *Add contact to Blacklist dialog* 37
4.18 *Blacklist with new entry* 37
4.19 *Remove contact from Blacklist dialog* 37
4.20 *Blacklist after removing* 37
4.21 *Main Screen* 38
4.22 *History screen* 38
4.23 *History details* 38
4.24 *Main Screen settings option* 39
4.25 *Spam detection enabled* 39
4.26 *Connect to the Server enabled* 39
4.27 *Select synchronization interval* 39
4.28 *Notification showed when the Blacklist is received* 40
4.29 *Connection to the server sequence diagram* 40
4.30 *Memory used* 42
4.31 *Storage size* 42

List of Tables

2.1 *OSI model adapted to the mobile VoIP network* 5
2.2 *Number of VoIP subscribers worldwide from the 3rd quarter of 2011 to the 2nd quarter of 2013 (in millions)* 6
2.3 *Estimated Fraud Losses due to Spamming by Region(in Millions $ USD)* 9
2.4 *Android OS evolution* 11
2.5 *Android OS use by version* 12
2.6 *Algorithm 1 - Reputation computation* 20
2.7 *Algorithm 2 - Detecting SPIT Caller* 21

4.1 *Blacklist searching time* 41
4.2 *Server time to compute the Global Scores and retrieve the Blacklist* 41
4.3 *Bandwidth used when exchanging information with the Server* 42

Abreviations and Symbols

CFCA	Communications Fraud Control Association
SPIT	Spam Over Internet Telephony
SP	Service Provider
OS	Operating System
VoIP	Voice over Internet Protocol
CDR	Call Detailed Records
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
RTP	Real-time Transport Protocol
IP	Internet Protocol
OSI	Open Systems Interconnect
GSM	Global System for Mobile Communications
UMTS	Universal Mobile Telecommunications System
WCDMA	Wideband Code Division Multiple Access
LTE	Long Term Evolution
CRC	Cyclic redundancy check
DoS	Denial of Service
SIP	Session Initiation Protocol
IETF	Internet Engineering Task Force
SNMP	Simple Network Management Protocol
IMS	IP Multimedia Subsystem
BTS	Base Transceiver Station
TLS	Transport Layer Security
UI	User Interface
App	Mobile Phone Application

Chapter 1

Introduction

"Spam over Internet Telephony (SPIT) is a challenge and can become a major threat when telephony completely shifts to VoIP"[7, p. 1].

The stand-alone spam detection systems can allow spammer to pass through the system lead "due to their few recipient within the service provider"[8, p. 1]. A way to improve spamming detection is to use a collaborative system where information can be exchanged among the service providers. However, privacy issues can arise to "the collaborating entities and their end-users"[8, p. 1].

Therefore, the following sections refer the motivation for this dissertation, the problem related to it and its contribution to the existing spam calls detection systems.

1.1 Motivation

In the 2013 Global Fraud Loss Survey, made by the CFCA [1], it is referred that the telephone fraud losses raised up to $46.3 Billion (USD) in 2013. Thus, it is urgent the creation of security mechanisms capable of reducing those losses and to mitigate telephony frauds.

On the other hand, the VoIP "provide cheap telephony service" which can lead "telemarketers, prank callers, and spammers" to "send bulk unsolicited calls"[6, p. 1, 4]. These calls can result in several issues, such as: threat against callee account credit; missing important calls; vishing; hijack of the network Equipment; mobile phone virus; negative perception about SP. These threats are better explained in the chapter 2, section 2.3.

Other important aspect is the fact that there is no mobile applications available that use the collaboration among their user for effective blocking of spammer.

1.2 Problem

Using the Caller-Rep algorithm, the data from numerous clients can be aggregated in order to improve the voice spam detection.

However, at this point, the CallerRep is constrained to voice operators, that is, it only works if there is some sort of log of CDR (Call Detailed Records) from several clients to which the operator can apply Caller-Rep.

Thus, the *App* tries to solve this problem providing to its users a way to share their data, independently from the voice operator, in order to start a cooperative service with the main goal of improving voice spam detection.

1.3 Contribution

After a research made in the Google Playstore, it was concluded that there is no applications developed for the Android OS that make use of the reputation that a certain caller has with other users in a network.

Therefore, the main contribution of the *App* is the fact that the blacklist is mainly constructed by a centralized server using the Caller-REP algorithm to decide which caller is spam or not spam.

Other contribution is that the incoming spam calls are blocked, which contributes to minimize the threats referred previously (see Fig. 1.1).

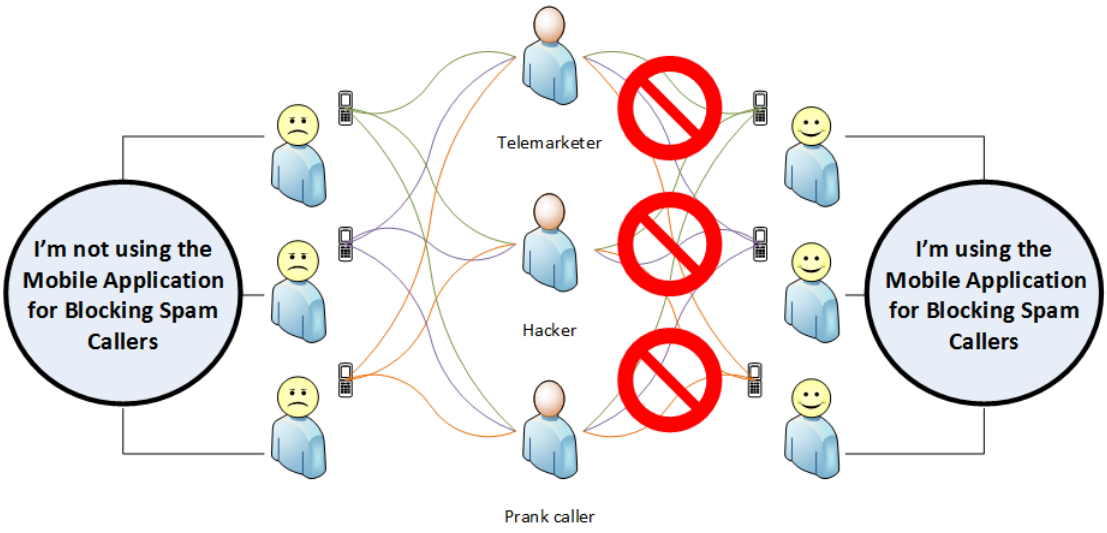

Figure 1.1: *Why using Mobile Application for Blocking Spam Caller?*

Throughout this report it will be described all the work done during the *App* development in order to achieve the proposed goals.

In the chapter 2, we first discuss the challenge of blocking the spam calls and provide some Technical Background on this subject.

Then in the chapter 3, it will be described the system architecture implemented and some Android OS API limitations. This chapter can be used to better understand the way the data flows throughout the system elements and how they communicate with each other.

The Implementation and Evaluation of the App are presented in the Chapter 4. In this chapter, we describe the System Class diagrams, the System Databases and some Use Cases related to the main Mobile Application functionalities.

To finish this document, we have the chapter 5 which refers the accomplished goals and makes a framework proposal that uses the Caller-REP algorithm and how it can be integrated in other voice calls applications like Viber, Skype, among others.

Chapter 2

Technical Background and Related Work

2.1 Technical Background

2.1.1 VoIP

As in an IP network, where messages are exchanged using datagrams, VoIP makes use of those to exchange voice data between users. Therefore, it needs a TCP/IP network to work [21, 10, p. 15-19, p. 10].

So, the OSI model can be used to describe the mobile VoIP networking, with its seven layers:

Session, Presentation and Application Layers	SIP
Transport Layer	TCP (reliable but slow) UDP (less reliable but fast) RTP (transfers audio between two or more endpoints)
Network Layer	Addressing scheme (IPv4, IPv6) Data routing
Data Link Layer	Quality of service functions (error check and CRC)
Physical Layer	Radio links(GSM, UMTS/WCDMA or LTE)

Table 2.1: *OSI model adapted to the mobile VoIP network*

In the following table, we can observe the continuous growth of the VoIP technology usage[1]

Quarter	Subscribers number (in millions)
3rd '11	131.73
4th '11	135.37
1st '12	140.31
2nd '12	144.1
3rd '12	147.69
4th '12	151.53
1st '13	155.17
2nd '13	158.7

Table 2.2: *Number of VoIP subscribers worldwide from the 3rd quarter of 2011 to the 2nd quarter of 2013 (in millions)*

2.1.1.1 SIP

An user, in order to make a VoIP call, must have an infrastructure that has some kind of protocol to allow him to use the network, to which he is connected, to make that call. Thus, one of the most used protocols is the Session Initiation Protocol (SIP).

This protocol is a standard defined by IETF and it works at the application-layer control and is used to "establish, modify, and terminate multimedia sessions (conferences) such as Internet telephony calls" [19, p. 9]. In a first moment, there is an exchange of SIP messages, between the caller and the callee, to get the following informations:

> **User location**: determination of the end system to be used for communication;
> **User availability**: determination of the willingness of the called party to engage in communications;
> **User capabilities**: determination of the media and media parameters to be used. [19, p. 9]

Next step will be the:

> **Session setup**: "ringing", establishment of session parameters at both called and calling party. [19, p. 9]

During a call, the SIP can make the:

[1] Source: http://www.statista.com/statistics/236821/number-of-voip-subscribers-worldwide/

> **Session management**: including transfer and termination of sessions, modifying session parameters and invoking services. [19, p. 9]

The SIP User Agent (UA) and SIP Network Server are two entities that make part of the SIP core network. Thus, they have different roles in the VoIP communications:

SIP UA: Is the user soft or hard phone responsible for initiating and accepting calls.

SIP Network Server: Manages signalling sessions among participating entities and has three main functional components: the SIP Registrar, the SIP proxy server and SIP redirect server. [8, p. 3]

The model adopted by SIP is the request and response for session management among communicating entities. So,

> "the request messages are sent from the user to Registrar for registration, call request for the start of new session, updating the existing session parameters, acknowledging session establishment and terminating the existing sessions. Response messages are used for providing the appropriate reaction to the request messages, depending on the type of request message. The SIP network, in addition, also consists of other supporting servers such as: a CDR server for storing Call Detailed Record of their users call transactions, billing system for billing and presence servers for storing the location and status of user." [8, p. 3]

2.1.2 Advantages of VoIP

Due to its nature, VoIP offers the cheapest way to make phone calls. Therefore, its wide adoption in home and business networks has being grown. Since it is based on a IP network, a VoIP network is simpler to build because the elements needed by it are already available, most of the times.

In business environment, there's no need to invest in a dedicated infrastructure for voice-only purpose and the one used for the network is enough to configure the VoIP service, allowing the users to make VoIP calls.

Another advantage is that there is only one network to maintain. Therefore, the administrator maintenance concerns rely on that network [21, p. 14].

2.1.3 Network concerns for VoIP services

With VoIP, data and voice have the same physical path. So, the VoIP infrastructure configuration must be done carefully and there must be some sort of design guidelines in the data network construction where it will work. Since it is an application that works in real time, the reliability and stability of the overall system must be taken in account, because with an equipment or energy failure the service can become unavailable [21, p. 14].

Other aspect that is important in the VoIP usage is the security and privacy that must be provided to its users. Therefore, mechanisms and techniques to fight against possible VoIP network threats must be provided.

2.2 VoIP Security Threats

VoIP security attacks can be divided in two kinds: active and passive attacks.

Active attacks are related to modifications to the VoIP infrastructure that results in some sort of DoS. These kind of attacks can be achieved through the exploitation of the weaknesses of its components.

These attacks can be accomplished by:

- Getting or uploading a VoIP hard phone configuration file - gets access to hard phone's settings and options or makes upload of an altered configuration file to make changes in the settings of the phone or control it remotely;
- Exploiting weaknesses of SNMP - Gain access to hard phone's configuration settings, if SNMP is enabled and using SNMPv1 read and write community strings;
- Impersonating VoIP devices - Spoof legitimate gatekeeper, Registrar, Proxy Server [10, p. 113 - 126].

Passive attacks are those which involve disclosure of sensitive or private information or misuse of the VoIP infrastructure to make unauthorized calls or flooding it with repeated calls.

The attacker can achieve his goals with, for example:

- VoIP phishing (*vishing*) - Fake phone number or phone destination to get private information;
- Caller ID spoofing - posing by a legitimate phone number;
- Anonymous eavesdropping/call redirection - Either using Caller ID spoofing or a phishing email;
- SPIT - Bulk advertisement through the VoIP network [10, p. 131 - 152].

2.3 Spam over Internet Telephony (SPIT)

SPIT is used to designate unwanted, automatically dialled, pre-recorded phone calls using VoIP infrastructure. [7]. Therefore, there are some aspects, related to the nature of SPIT, to be taken in account when fighting against it, such as:

- A voice call is done in real time;
- Its contents are only available when the call is established; [8, p. 3].

- There no distinguishing header for either SPIT or non-SPIT caller, therefore it can't be used to detect SPIT calls;
- The space required for a voice message can unable legitimate callers to use the resources associated to their voice mailbox.

The estimated fraud losses due to Spamming by Region can be observed in the following table[2]:

Region	Losses (in millions)
Asia	70 000
South Pacific	20 000
Central and South America	20 000
North America	230 000
Western Europe	250 000
Eastern Europe	90 000
Africa	70 000
Middle East	50 000

Table 2.3: *Estimated Fraud Losses due to Spamming by Region(in Millions $ USD)*

2.3.1 SPIT Threats and Scenarios

SPIT has many scenarios of applicability, depending on the nature of the SPIT call, such as:

1. **Threat against Callee account credit** - loss of callee account due to call forwarding, roaming and automatic call back service when receiving a value added service call, like an advertiser call;
2. **Threat against missing important calls** - when the subscriber voice mailbox is enable, a spit call diverted to it can result in making that resource unavailable to the legitimate callers. When unsolicited calls are in great number, the time to delete them from the voice mailbox can be a "time consuming job"[8, p. 4], because the callee must listen everyone of them, for a little bit of time, before deleting each one;
3. **Vishing** - hiding the identity or impersonating legitimate identities to get private information leading to disclosure of callee private information;
4. **Network Equipment Hijacking** - when attacker gets access to a VoIP network element and use it to send unwanted calls;

[2]Source: 2013 Global Fraud Loss Survey(CFCA[1])

5. **Mobile Phone Virus** - the attacker sends virus through unsolicited communications, in-order to, for example, destroy the operating system of the IP phone or steal the user contact list.

2.4 Android

2.4.1 Brief History and Evolution

Android is an mobile operating system mostly used in smartphones and tablets.

However, with the new API 20 [2], there's a new generation of devices emerging, such as: wrist watches, Smart TVs and auto devices. [3, 4]

This mobile OS was founded by Andy Rubin, Rick Miner, Nick Sears and Chris White in Palo Alto, California. Later, Google bought it in July 2005. Thus, in November 2007 they made an public announcement that they were developing their first 'Google Phone' built on top of a mobile platform, Android, with a Linux based kernel.

According to Andy Rubin "Android is the first truly open and comprehensive platform for mobile devices. It includes an operating system, user-interface and applications – all of the software to run a mobile phone, but without the proprietary obstacles that have hindered mobile innovation"[3].

Android was developed by Open Handset Alliance in aliance with others technology and mobile leaders that includes: Motorola, Qualcomm, HTC and T-Mobile. Andy Rubin refers that "Through deep partnerships with carriers, device manufacturers, developers, and others, we hope to enable an open ecosystem for the mobile world by creating a standard, open mobile software platform"[3].

Therefore, "Android is a fully open and comprehensive platform ... that will give mobile operators and device manufacturers freedom and flexibility to design products and programs"[4].

In the Table 2.4 there are presented the several Android OS versions[5] and their respective first release dates.

[3] Source: http://googleblog.blogspot.pt/2007/11/wheres-my-gphone.html
[4] Source: http://www.ibtimes.com/evolution-android-os-g1-jelly-bean-697079
[5] Source: http://developer.android.com/tools/revisions/platforms.html

Version	Release date
Android 4.4W	June 2014
Android 4.4	October 2013
Android 4.3	July 2013
Android 4.2	November 2012
Android 4.1	June 2012
Android 4.0.3	December 2011
Android 4.0	October 2011
Android 3.2	July 2011
Android 3.1	May 2011
Android 3.0	February 2011
Android 2.3.4	May 2011
Android 2.3.3	February 2011
Android 2.3	December 2010
Android 2.2	May 2010

Table 2.4: *Android OS evolution*

In the Fig. 2.1 and in the Table 2.5, it can be observed how the Android OS versions usage is distributed:

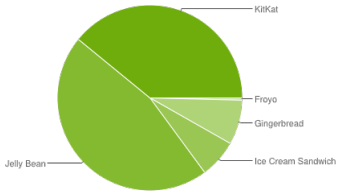

Figure 2.1: *Smartphone OS Market Share*

The information available in the Table 2.4 is based on the "data collected during a 7-day period ending on July 7, 2014"[6].

[6]Source: http://developer.android.com/about/dashboards/index.html

Version	Codename	API	Distribution
2.2	Froyo	8	0.7%
2.3.3 - 2.3.7	Gingerbread	10	13.5%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	11.4%
4.1.x		16	27.8%
4.2.x	Jelly Bean	17	19.7%
4.3		18	9.0%
4.4	KitKat	19	17.9%

Table 2.5: *Android OS use by version*

2.4.2 Market Share

According to the IDC Smartphone OS Market Share[7] there has been a clearly growth of the Android OS use, as IDC states that:

> "**Android** proved once again it is positioned where the market is going by growing its volume from the fourth quarter, something that doesn't happen too often given the smartphone market's seasonality. Larger 5-7" Android devices grew to 84.5 million in 1Q14, which was 36.2% of all Android shipments globally. Samsung leads in the Android camp, while the rest of the pack is quickly being made up of Chinese vendors. Huawei, Lenovo, Coolpad, Xiaomi, ZTE, and OPPO were all part of the top 10 Android vendors in the first quarter of 2014" (IDC, 2014 Q1).

2.5 Anti-Spit Overview

In real time communications, such as VoIP or mobile telephony, the detection of spam (is this case spam calls) is much more difficult, because a real-time response is needed for the users call requests, making the VoIP calls spam detection different from the email spam detection.

In RFC5039 [18] several approaches are shown to block SPIT callers, that can be divided into two categories:

1. Methods based on content analysis of signaling messages and actual voice (transparent to the caller and the callee);
2. Methods that require feedback from the caller or the callee about the spamming nature of the call. [6, p. 5]

[7] Source: http://www.idc.com/prodserv/smartphone-os-market-share.jsp

According to how the decision (if a call is or is not spam) is made, there can be two types of methods:

1. Pre-acceptance - Decision is made before the call is answered;
2. Post-acceptance - Decision is made after the call is answered.

Based on the assumptions previously enumerated, several approaches emerged to fight against spam calls, which the main characteristics and issues will be described in the following subsections.

2.5.1 Content-based approaches

This type of approaches has the principal aim to analyse the "semantics of SIP signaling messages or the contents of the RTP stream" [6, p. 5], which can be done either in real time or in non-real time.

Thus, when real-time voice processing is used that demands real-time signal processing, which can lead to a deteriorated voice quality, due to the voice processing delay.

Another important aspect is that there is an issue related to privacy, because "contents are encrypted and background noise is added to the speech" [6, p. 5] leaving the data vulnerable to disclosure.

With content-based techniques there's no additional information added to calls, so it becomes hard to make the distinction between the SPIT caller and the non-SPIT one.

2.5.2 Access list-based approaches

A list of databases are used, in list-based approaches, to check the identities of VoIP users [15, 9] There is an use of blacklists, which store the users that must be blocked from calling, and whitelists, containing the identities of the users allowed to make calls. This blocking method requires frequent updates of both lists, to maintain its usability.

A third list can be used, usually called gray list. The first time a user, contained in the gray list, tries to make a call that is not allowed. When he tries to do a second call, that is allowed whithin a defined time window. Thus, this implies "multiple attempts to reach the callee" [6, p. 6].

Usually, this techniques can be combined with others [16, 17, 14].

2.5.3 Challenge response-based approaches

A Turing test can be used to distinguish human from machine calls, because the SPIT calls can have its origin in automated machines. The use of this tests "is based on the fact that humans can easily solve some problems which are impossible for the computer" [6, p. 6].

Although computer generated SPIT calls can be blocked using the Turing test approach, it requires a lots of network and computational resources. This approach can annoy users with its puzzles, because they must solve one in every call they make leading to a larger call setup time.

2.5.4 Imposing additional costs on callers

This type of approaches, based on Payments at Risk [22], firstly takes off a small amount of money from the caller account, then returns it, in a later step if the caller is proved to be legitimate.

The implications of this mechanism are:

1. Some sort of feedback from the callee or content processing is needed to make the final decision;
2. There should be a balanced payment system.

Taking into account the first requisite, it is not easy to implement because callers are loath "in providing feedback for every received call and content processing faces same limitation of content based approaches" [6, p. 6].

Due to the different call setups, it is hard to design a payment system which is convenient to each one of them.

2.5.5 Extended call-setup based approaches

This type of approach uses the proxy server that takes the following steps:

1. Accepts the call;
2. Disconnects the call;
3. Calls the caller back (he must be able to pick up the call).

However, some limitations can be appointed to this approach such as need of additional network resources and increased call setup time.

2.5.6 Social reputation-based approaches

VoIP users are ranked according to the social relationship between each other. Thus, this is done in two steps:

First step: calculate the trust value, "computed between any two users" [6, p. 6], which can be done in two ways: either attributing positive or negative feedback to a caller [16, 17, 23, 12, 13] or calculating the average call duration of the calls made to a certain callee [20, 22];

Second step: calculate the global reputation, which provides "an indication of the spamming behaviour of a caller in the whole network". This reputation can "be computed from social network features including node degree, local clustering coefficient, in-count degree, out-count degree, reciprocity index etc" [6, p. 6].

Based on this steps, there are several social reputation-based methods, such as:

1. CallRank [20] - gets direct trust through average call duration; uses Eigen trust reputation algorithm to get the global reputation. The higher the average call duration is, the higher is the "trustworthy relation between caller and callee" [6, p. 6] and it is used to user reputation across the network;

2. Semi-supervised clustering to callee feedback and to the distribution of SIP messages [23] - groups callers into legitimate and non-legitimate clusters; this approach requires user feedback and changes to VoIP software.

3. Multi-stage SPIT detection system consisting of trust and reputation stages integrated with black and white lists [16, 17] - computes the trust based on callee feedback, either negative or positive. Bayesian inference is used to build the caller reputation.

4. Reputation based techniques in combination with other SPIT detection approaches [22, 7] - multistage method and "interact with other stages for a final decision about the nature of the caller" [6, p. 6].

5. Three SPIT detection techniques based on average call duration, degree distribution and reciprocity index [11] - caller reputation among network users is calculated using the average call duration with a page-rank algorithm.

6. Two computers systems based on the entropy of the average call duration [12, 13] - system number one distinguish SPIT from non-SPIT callers through Mahalanobis distance to call duration and time of call. System number two detects misbehaving groups based on the entropy of call duration at a group level.

7. Collaborative score-card framework [5] - used in IMS network to distinguish legitimate caller from the non legitimate and is based on the exchange of score-cards of a caller within the receiving domain, which are used to block or allow a call.

8. Multistage SPIT detection system [7] - the detection of the "SPIT caller in a transit VoIP operator" [6, p. 7] is made using a feedback between several stages.

2.6 Stand-alone System Spam Detection

This kind of systems are widely used for detecting and blocking SPIT caller in a VoIP architecture. They consist, generally, in the observation of local traffic or call patterns to decide if a call is or isn't spam.

The stand-alone systems can be implemented using two distinctive ways:

- Content based approaches - as the call contents are only available after call setup, is hard to implement in a VoIP infrastructure. This needs real time speech processing and databases containing spam speech contents which require expensive resources, making it hard to deploy and not feasible;

- Identity based approaches - requires callee feedback about the caller or caller's call detailed records to be analysed to calcualte the reputation of a certain caller.

Smart spammer can easily hack standalone systems because these systems don't take in account the caller's calling behavior, making the system "sensitive to the detection accuracy" [8, p. 5]. However, it can be more robust against spammer attacks "when implemented in the form of multistage system" [8, p. 5].

2.7 Social network features for VoIP users

The CDRs are stored in a server and can have several information about a call, such as: IDs of the caller and the callee, time and duration of the call, quantity of packets transferred in both directions, which party disconnected the call, among others.

Thus, this CDRs can be used to model the caller-callee social network, which can be represented as a directed call graph G=(V,E).

Specific calling patterns are observed when legitimate VoIP callers make a call to his friends or family members, which form his social network.

However, when a telemarketer or spam caller makes a call, their calling patterns and social network are different from the legitimate callers. Therefore, these "social calling behaviors" [6, p. 7] can be used to gather information to identify spam callers.

To make the identification of a spam caller it will be used the social network features described in the next three subsections.

2.7.1 In-degree and out-degree

- In-degree of a user - number of other unique users calling this user and is defined as follows:

$$In - Degree(S_i) = \sum_j E_{ij} \tag{2.1}$$

- Out-degree of a user - number of unique users this user calls and is defined in the equation 2.2:

$$Out - Degree(S_i) = \sum_j E_{ij} \tag{2.2}$$

E_{ij} "is one if user i has called user j at least once, and zero otherwise" [6, p. 7].
The following patterns can be observed:

- SPIT caller - makes lots of calls to a large number of unique callees but the answers he receives are only few, leading to an unbalanced in/out-degree;

- Legitimate callers - get balanced in/out-degree because the number of calls they make and receive are almost the same.

However, using these in/out degrees in a isolated way may result in a low detection rate because SPIT callers can often change their identities.

2.7.2 Call rate

Defined as the sum of the number of calls made or received by a caller that can be categorized as in-call and out-call rates.

This rate reflects the behaviour of an user: "the higher the call rate, the more frequently a user calls the same people" [6, p. 8]. On the other hand, "SPIT callers try to call as much people they can" [6, p. 8], which "results in a small calling rate and non-repetitive behaviour" [6, p. 8].

2.7.3 Call duration

Defined as the "total duration of all calls made or received by the user" [6, p. 8], which can be classified as in-call and out-call duration.

High duration calls are observed in calls made by legitimate callers "within their social groups" [6, p. 8], while, out of their social groups" [6, p. 8], the duration is less.

The principal target of SPIT callers are new callees (previously unknown to them) and when a callee receives a call from a SPIT caller, usually he doesn't know the number, so he decides "the nature of caller within the first few seconds of conversation" [6, p. 8], and disconnects the call after those seconds. This leads to a short time of conversation, implying that the SPIT callers are associated to a large number of short duration calls, which differentiates them from "socially connected legitimate callers having large duration calls" [6, p. 8].

2.8 Caller-REP

2.8.1 Why Caller-REP?

Developing a SPIT detection system which blocks calls during the signaling phase and without content analysis or user involvement can be challenging.

Because of the repetitive calling behaviour that a legitimate caller has with their friends and family members (Bokharaei et al., April 2011), it is possible to determine the trust and reputation "from the average call duration along with social network features" [6, p. 7], by using, in a collective way, "the number of repetitive calls, the number of reciprocal calls, call duration in both directions, and the number of unique callees" [6, p. 7].

So, in a first step, the direct trust is computed using "the number of outgoing partners, the calling rate in both directions, and the total call duration in both directions" [6, p. 7]. On a second step, the global reputation is calculated and "the caller is classified as legitimate or non-legitimate using an automatic threshold approach applied to the caller reputation" [6, p. 7].

Caller-REP consists in, first, using "a combined set of caller's social network features" [6, p. 7], as described in the previous subsections, for computing direct trust between VoIP users.

Second, the power iteration method is used to calculate the caller's reputation. Third, in order to make the final decision the 25th percentile values are used to compute an automatic threshold.

Caller-REP analyses the "importance of relationships between users and the behaviour of users across the network" [6, p. 8]. Thus, it has three steps to categorize users as SPIT or non-SPIT:

1. "Extracts the social network of the caller from the CDR data and computes direct trust between a user and their callees" [6, p. 8];
2. "Uses the power iteration algorithm for computing the caller's reputation across the network" [6, p. 8];
3. "Computes a threshold for the automatic classification of caller as SPIT and non-SPIT" [6, p. 8].

The following figure represents the block diagram of Caller-REP.

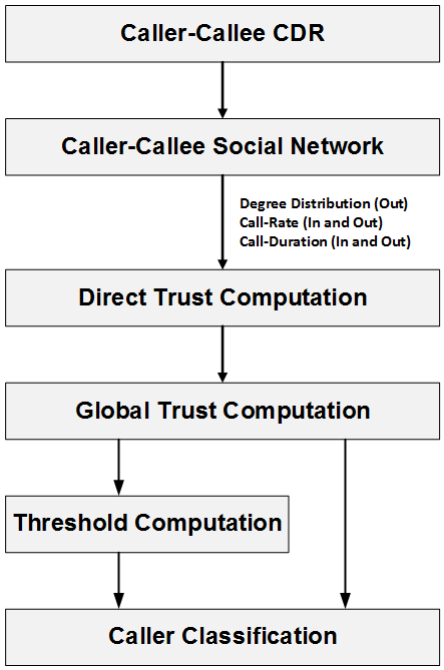

Figure 2.2: *Caller-REP block diagram*

It is used the past interactions between a user and the others users to make his social network, using the following features:

"Call Duration: Talk time of a user. CD_{SR} is the sum of the duration of all the calls made by user S to user R;

Call Rate: Number of calls made and received by the caller. $CallRate_{SR}$ is the number of calls user S made to user R;

Partners: Number of unique callees associated with each user in a network. PO_S is the number of unique callee of caller S and PI_S are the number of unique callers calling caller S." [6, p. 8]

Then, the callers direct trust is computed to all his partners, which is used to calculate the callers global reputation and the automatic threshold.

In the next subsections are described how the caller direct trust, the caller global reputation and the automatic threshold are computed.

2.8.2 Caller direct trust

"Direct trust between caller and the callee is the combination of the amount of time both users are engaged in talking, the number of reciprocal calls between them, and the number of unique callees of the caller" [6, p. 8-9]. This parameter measures the social relationship between a caller and a callee.

It is important to refer that a "legitimate caller usually has strong social ties with a large number of callees and weak ties with a few called callees" [6, p. 9]. However, a SPIT caller develops "weak social ties with a large number of called callees" [6, p. 9].

Thus, the following features are used for "the computation of callers S direct trust score with callee R", which can be observed in Equation (2.3):

1. The out-going number of partners of the caller, PO_S;
2. The number of out-going repetitive calls $CallRate_{SR}$ and their call duration CD_{SR};
3. The number of incoming calls $CallRate_{RS}$ and their call duration CD_{RS}.

.

$$TrustSR = \frac{CD_{SR} \times CallRate_{SR} + CD_{RS} \times CallRate_{RS}}{PO_S} \tag{2.3}$$

2.8.3 Caller global reputation

Caller global computation is computed after the callers direct trust, which represents "his reputation across the network" [6, p. 9]. This has an important role "when the callee receives a call from an unknown caller and relies on the collaborative feedback of other callees that already interacted with the caller" [6, p. 9].

Thus, the caller reputation across the network is based on the trust scores he has with the callees, the higher the trust score is the higher is the reputation and the lower the trust score is the lower is the reputation score.

Using the Algorithm 1, presented on the Table 2.6, it is possible to calculate the reputation of a caller.

The matrix of normalized direct trust T_{SR} between every pair of users (S,R) is the input of this algorithm, while the output is "a global reputation vector G with a per-user reputation score $G_S \in [0,1]$" [6, p. 9].

1:	**procedure** GLOBAL REPUTATION OF ALL USERS
2:	$input \leftarrow T$ (normalized direct trus matrix, with elements T_{SR})
3:	$output \leftarrow GR$ (Global reputation score vector, with wlements GR_S)
4:	$precisionparameter \leftarrow \varepsilon$
5:	Initalize reputation vector GR
6:	$GR_S = \frac{1}{PO_S}$
7:	**while** $\delta < \varepsilon$ **do**
8:	$GR \leftarrow T \times GR$
9:	$GR \leftarrow \frac{GR}{\|GR\|}$
10:	$gr \leftarrow \| GR \|$
11:	$\delta \leftarrow \frac{gr - gr_{previous}}{gr}$
12:	$gr_{previous} = gr$
13:	**end while**
14:	**end procedure**

Table 2.6: *Algorithm 1 - Reputation computation*

Then, an iteration occurs and finishes only when the "convergence of the norm of the global reputation vector $\|GR\| = \sqrt{\sum_S GR_S^2}$ " [6, p. 9] is reached. In this iteration occur several steps:

1. Global reputation vector is updated ($GR = T \times GR$);

2. GR is normalized;

3. Norm gr is checked for convergence with previous norm $gr_{previous}$

2.8.4 SPIT caller detection

Caller-REP uses a dynamic threshold learned from the set of reputation values. Therefore, the dynamic threshold value is based on a percentile method that make use of the "set of computed reputation scores of all callers" [6, p. 10] and corresponds "to the 25th percentile of this set" [6, p. 10].

The algorithm described on Table 2.7 is used to classify callers as SPIT or non-SPIT.

The value m represents the 25th percentile value of the global reputation which is computed for each time window. The dynamic threshold is set based on the mean of the callers with global reputation less than m (line 6).

Callers are classified as legitimate 1 or non-legitimate -1 based on a following rule:

$$Callers_S = \begin{cases} GR_S > \beta\ threshold; & 1 \\ GR_S < \beta\ threshold; & -1 \end{cases}$$

To "maximize true positive and true negative rates in a network with high SPIT or legitimate traffic, Caller-REP can be implemented with a β parameter that is defined by the operator for controlling false detection" [6, p. 10].

```
1:  procedure SPIT DETECTION
2:      input ← Global Reputation(GR), with elements GR_S
3:      output ← SPIT(1) or non-SPIT(-1) detection vector, with elements SPIT_S)
4:      operator − defined parameter ← β (β = 1 if operator has no preference)
5:      m ←1st-quartile(GR)
6:      threshold ← mean(GR < m)
7:      for All caller S do
8:          if (GR[S] < β × threshold) then
9:              Place Caller S in a SPIT list
10:         else
11:             Do Not Place Caller S in a SPIT list
12:         end if
13:     end for
14: end procedure
```

Table 2.7: *Algorithm 2 - Detecting SPIT Caller*

Using this type of SPIT detection implementation, that makes use of the social network features like Caller-REP, there can be developed a system for SPIT detection independent from the cellular/VoIP operator.

The application developed uses this approach in order to avoid operator constraints concerned to the share of its information about spam callers. Therefore, it constructs the blacklist using the social reputation that each user has among the other users in the system and doesn't need to use the operator list to do so or any information manually given by their users.

Chapter 3

Spam Blocker Architecture

3.1 Application Elements

The system developed is based on the Caller-REP SPIT detection described in section 2.4. However, it will handle only cellular calls.

In the next sections, it will be explained how the reputation is calculated, how the reputation scores is exchanged between the system components, how a call is handled by the Android OS and how a call is blocked by the application.

3.1.1 System Architecture

Fig. 3.1 shows the architecture of our proposed solution for detecting spams

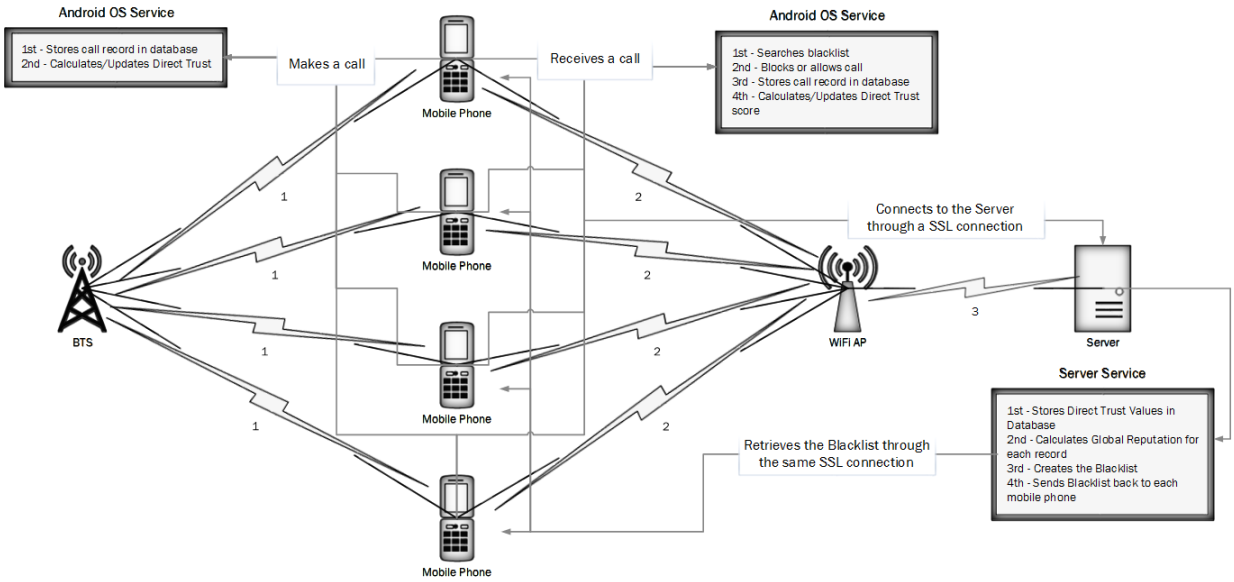

Figure 3.1: *System Architecture*

Below I briefly describe the functionality of each device in context with collaboration and spam detection.

3.1.1.1 Mobile Phones

Use the Android application to listen calls made or received, through the Android OS service, and process the information related to them. Thus, when an incoming/outgoing call is received or made, the application gets the phone number and gets duration of call from call logs when call get disconnected. With this information, the App computes the Caller Direct Trust using the equation 2.3 and records the result into the private database. The Caller Direct Trust values are then sent to the Server when a WiFi connection is enabled and the user connects to it.

3.1.1.2 Server

Server gets the direct trust values from the Mobile Phones (users), stores them in the server database and organizes them into a matrix to compute the *Caller Global Reputation* using the Algorithm 1 (See Table 2.6). Then, the list of spam callers is built based on the results got in the previous step and using the Algorithm 2 (See Table 2.7). The caller spam list (blacklist) is sent back to the users. The connection between the users and the server is made using a SSL socket. All users contribute with Direct Trust values to compute the blacklist in a collaborative manner.

3.1.1.3 BTS

BTS permits the mobile phone to receive and make mobile calls through the connection 1.

3.1.1.4 Wifi AP

Wifi AP allows the mobile phone to send the Direct Trust file to the server and receive the updated Blacklist from it through the connection 2 and 3.

In Fig. 3.1 it can be observed that there can be several users using the App and connecting to the server to exchange their Direct Trust files. Thus, it's important to refer that the system uses those Direct Trust files to compute the Reputation Score that a certain callee has among the system users. This Reputation Score is used to decide if that callee is or is not a spammer. The Blacklist is made taking in account that decision and then distributed to the users connected to the system server.

3.1.2 Call flow

In the following sections it will be described the *App* behaviour when the blocking feature is on and off.

3.1.2.1 With Call Blocker

Firstly, it is created a service to listen to incoming/outgoing calls and the Blacklist is copied to an hash table. Then, when a call is perceived, the hash table is checked to see if the callee is spammer or not. If he is a spammer, he is blocked and the call is terminated. Otherwise, the call is allowed to be established by the Android OS.

3.1.2.2 Without Call Blocker

Nothing is done and the Android OS handles the incoming/outgoing calls as usually.

3.1.3 Server Connection

In the next sections there will be explained how the data flows from the user to the server and vice-versa.

3.1.3.1 From user to Server

When the user enables the connection to the Server, first the *App* checks if an Wi-Fi connection is available. If there is one, then an SSL connection is created and the *Direct Trust* records file is sent to the Server. Then, the Server processes the file, stores the *Direct Trust* values in its database, and calculates the *Global Reputation* for the callees present in the file.

3.1.3.2 From Server to user

After computing the *Global Reputation* to each callee, the Server constructs the spammer list. This list is retrieved back to the user through the same SSL connection.

3.2 Constraints of the Android OS API

There was found some constraints in the Android OS API, which influenced the way the *App* was developed, which are:

1. **Listen VoIP calls** - the Android OS API does not permit to do it without having a SIP server configured by the user;
2. **Listen other Apps Voice Calls** - The Android OS API doesn't have a way to listen to voice calls from other apps like Skype, Viber, or other communication app. But if there are ways to do so, we didn't find any in order to solve this problem.

Chapter 4

Implementation and Evaluation

4.1 System Class Diagrams

4.1.1 Mobile Application Class Diagram

The classes used to create the Mobile Application were divided into three packages. However, only the one used to make the UI are represented here. Thus, the Fig. 4.1 contains all the classes related to the UI. Since these classes make use of others contained in the other packages, those will be explained when needed.

The MainActivity class represents the main screen that the user sees when the application starts. This class makes the transition to the others UI screen classes. These transitions are represented by the Intent objects and the ImageButton objects allows the user to select each screen he wants to go to.

The method onCreate() is used when an activity is launched and it is here where the objects are created and initialized.

The method onCreateOptionsMenu() is used to display the Menu Options, which can be different on each activity.

The method onCreateOptionsItemSelected() is used to handle the option that was selected by the user.

Since the classes BlacklistMain, ContactsMain and HistoryMain are also activities, the three methods explained above were also used but not included in the class diagram.

In the first time the App is used, the Calls Log of the Android OS is processed to get the Direct Trust to each contact recorded in that file. To do that, the MainActivity uses three methods:

1. loadContacts() - method from the DatabaseHandler class which is responsible to copy the contacts recorded on the mobile phone to the App private database;

2. getContactsNotInPhone() - method from the DatabaseHandler class which is responsible to copy the contacts not recorded on the mobile phone but recorded in the Call Log to the App private database;

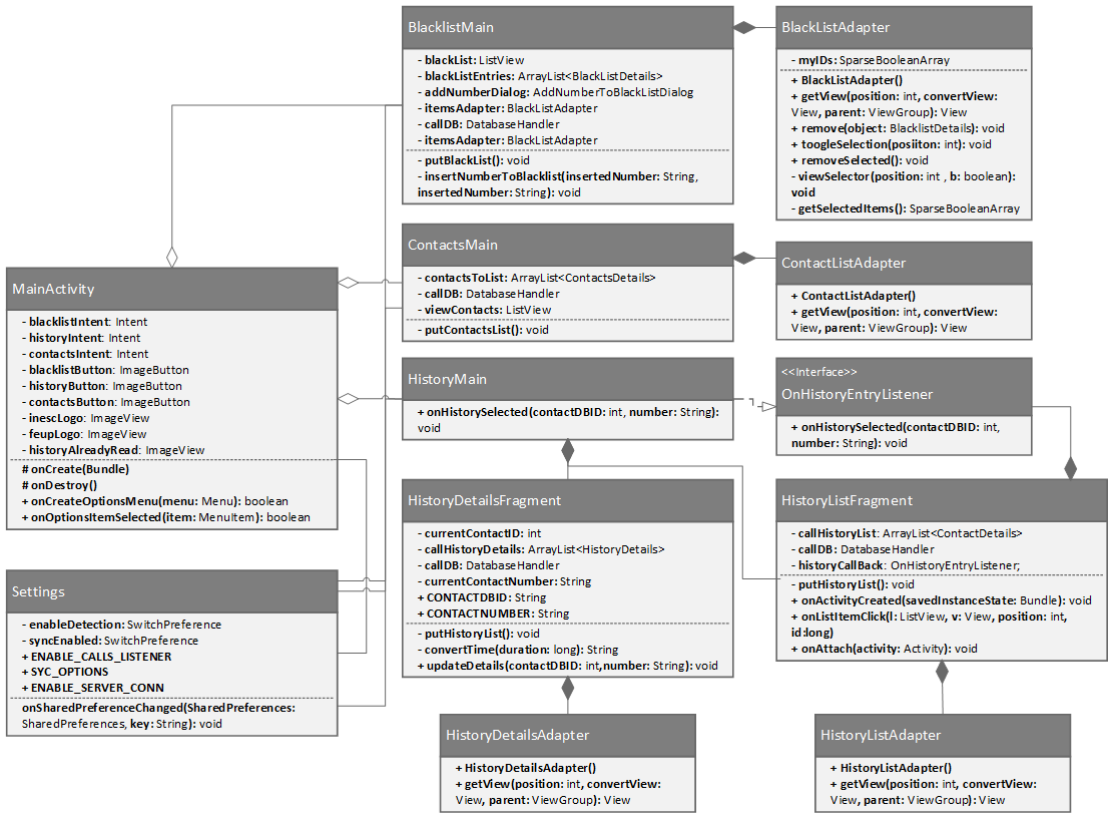

Figure 4.1: *Mobile Application Class Diagram*

3. readAllHistoryFirstTime() - method from the CallLogsProcessing class that computes the Direct Trust to all call records contained in the Calls Log;

The DatabaseHandler and CallLogsProcessing classes have different developed methods according to each activity functionality. Thus, the methods needed for each activity will be referenced throughout this subsection. The classes and their methods are represented in the figures 4.2 and 4.3.

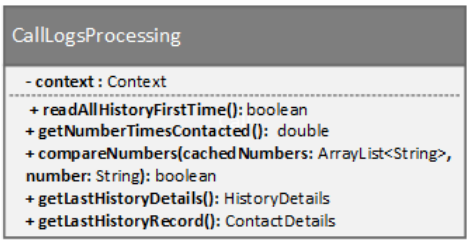

Figure 4.2: *CallLogsProcessing Class*

Most of the functions related to the contact numbers information storing and reading are associated to the DatabaseHandler class.

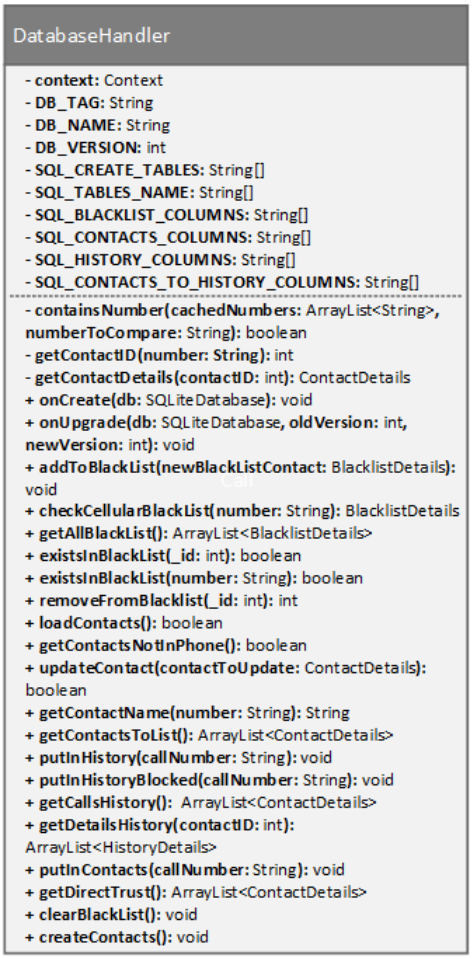

Figure 4.3: *DatabaseHandler Class*

The CallLogsProcessing class has the main function of processing the Calls log of the Android OS.

The class BlacklistMain permits the user to see which numbers are being blocked by the app.

The object that allows the listing of the blacklisted numbers is the *blacklist*. To get the information to being listed, it is used the object *callDB* to access the app database calling the method getAllBlackList(), which retrieves a list of BlacklistDetails objects. The BlacklistDetails class can be viewed in the Fig. 4.4.

To allow the removal of an entry from the Blacklist, it was needed to create the BlackListAdapter class, which is responsible for the layout of each entry of the *blacklist* and to handle long pressing touch on it. Thus, when the user decides to delete an entry, the *callDB* object calls the method removeFromBlacklist() to delete it from the database. The user can add manually numbers to the *blacklist*. For that he must click the icon presented on the menu and showed by the method onCreateOptionsMenu(). The *addNumberDialog* is an object of the class AddNumberToBlacklistDialog which gives the user the dialog needed to insert the name and number he wants to block. That class is represented by the Fig. 4.5.

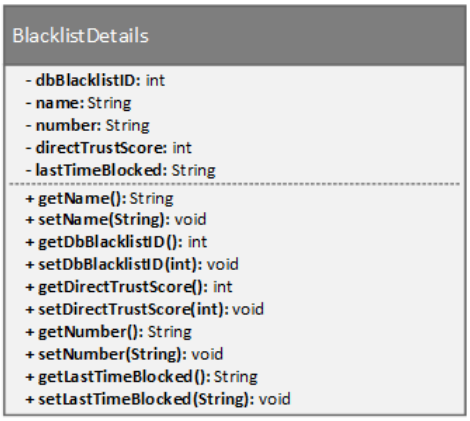

Figure 4.4: *BlacklistDetails Class*

This class has an interface that is implemented by the BlacklistMain activity to obtain the data inserted by the user. Then, after the dialog disposal, the data is inserted to the database using the callDB object by calling the method addToBlackList().

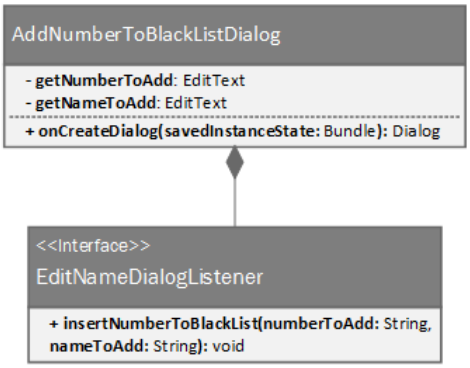

Figure 4.5: *AddNumberToBlacklistDialog class*

The ContactsMain activity lists not only the contacts that are in the mobile phone Contact List but also the numbers that are only in the Call Logs of the Android OS. This decision was made based on the fact that any number handled by the App should have a direct trust score. Thus, those numbers should also be stored into the App database, making the App to have its own contact records. Therefore, this activity displays all the numbers recorded in the database, providing the following information to each entry: name, number, last time contacted and direct trust. The contacts to display are stored in an ArrayList of objects of the class presented in the Fig. 4.6.

Then, the ContactListAdapter class is used to populate the ListView object (viewContacts) with the contacts info.

The calls history considers only the calls made or received when the call detection is enabled. Thus, the HistoryMain activity only displays those records. In this activity it was decided to use fragments instead of separated activities because this allows "to modify the activity's appearance

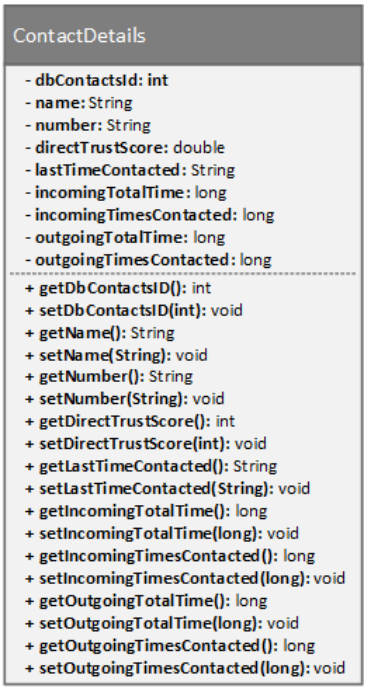

Figure 4.6: *ContactDetails class*

at runtime and preserve those changes in a back stack that's managed by the activity."[1]. This is an advantage because the user can click an entry to see the history details of a certain number and go back to the main screen without having long delays. Thus, this activity contains two fragments:

1. HistoryListFragment - displays the first information the user sees. It declares an interface used to handle the user entry selection. This interface is implemented by the main activity that retrieves the information captured by it to the next fragment.

2. HistoryDetailsFragment - displays detailed information about the entry selected by the user in the previous fragment. Each row shows the date, duration and time of each call.

The method to gather the information to display in the HistoryListFragment is getCallsHistory() and the one used by the HistoryDetailsFragment is getDetailsHistory(). Both fragments use an Adapter to show the data obtained with those methods.

Finally, the Settings class is used by the user to enable or disable the calls detection and to connect to the server to exchange the Direct Trust values with it and get the blacklisted numbers.

The classes used to make those functions work are represented by the Figs. 4.7 and 4.8.

To detect the incoming or outgoing calls, it was developed the CellCallsService class which runs in background when the user enables the calls detection feature. To detect the calls, it was needed the MyPhoneListener class to listen to phone state changes according to the type of calls being received or made.

[1]Source: http://developer.android.com/guide/components/fragments.html

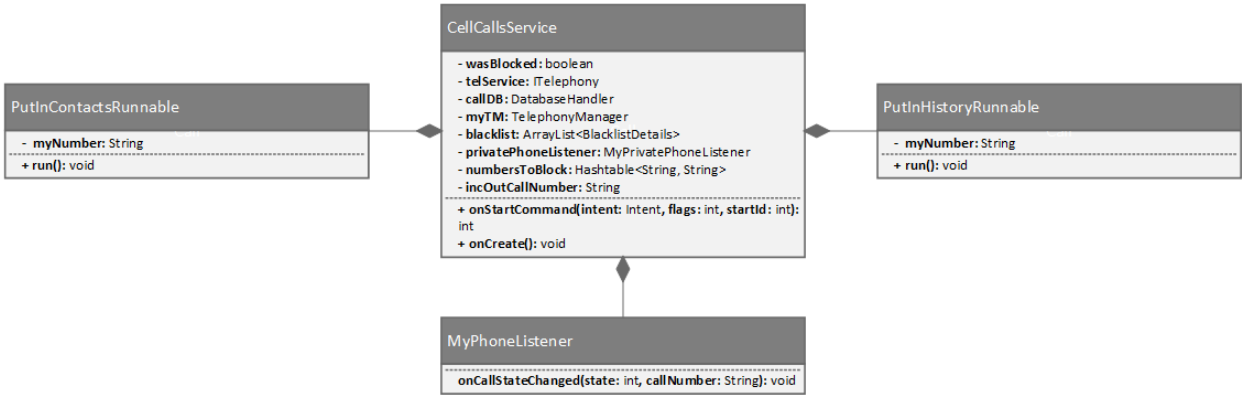

Figure 4.7: *CellCallsService class*

To update the database with the new calls records, there were created two threads using the classes PutInContactsRunnable and PutInHistoryRunnable. The first one has the purpose of adding the call number to the database if it is not recorded, the second put a new record to the database history table.

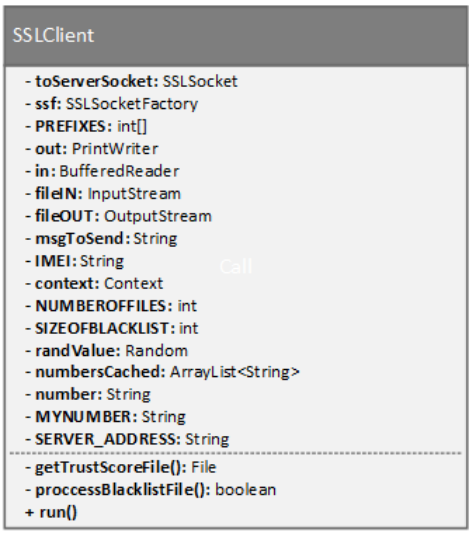

Figure 4.8: *SSLClient class*

The SSLClient class makes the connection to the server using the SSLSocket object to avoid eavesdropping of the information exchanged. After sending the Direct Trust file, using the method getTrustScoreFile which makes a copy of all Direct Trust values to a file, it is used the method processBlacklistFile to copy the spammer numbers from the file received to the mobile database.

4.1.2 Server Class Diagram

In this section it will be explained the role of the Server in the overall system. Therefore, the class diagram of the Server side is shown in the following figure:

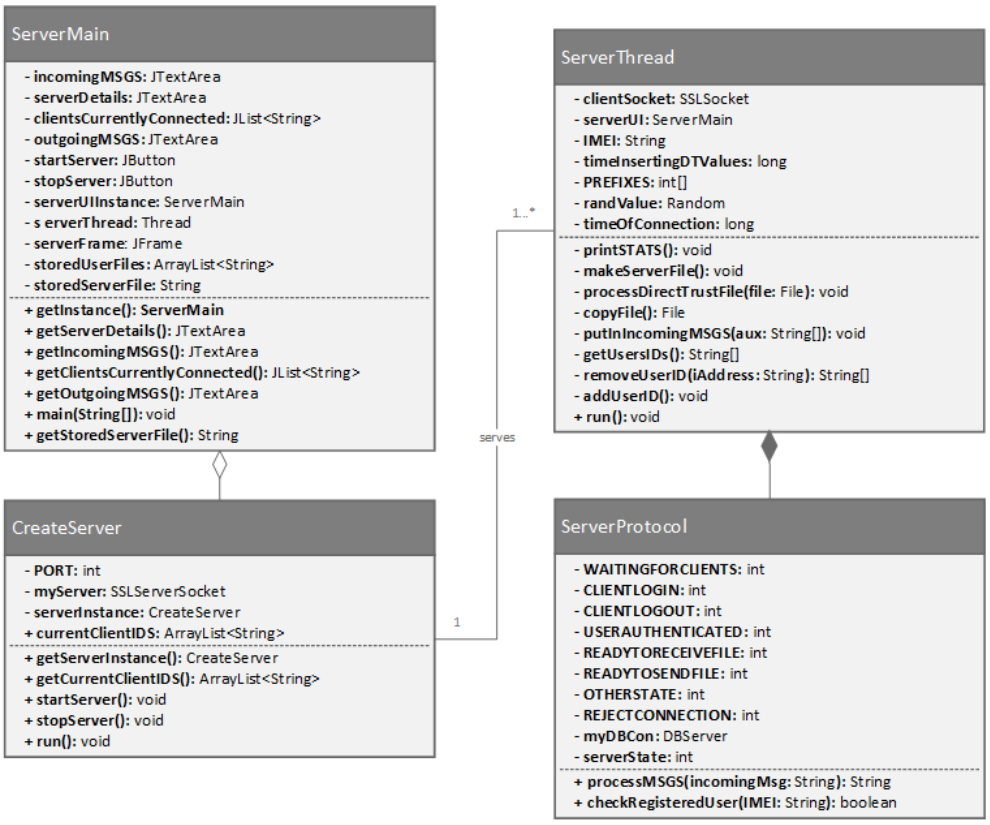

Figure 4.9: *Server side class diagram*

The ServerMain class is the UI that enables the system administrator to see the messages exchanged between the server and the mobile phone (client) of the user (see Fig. 4.10).

When the Server is started a CreateServer object is instantiated in a singleton manner. This guarantees that there is only one instance of the server. When a client connects to the Server, the CreateServer object creates an ServerThread to each client that tries to connect. Each one of them has an ServerProtocol object that is used to process the incoming messages from the client.

The connection to the server is made by an SSL socket. Thus, when a device is not registered in the system, it is automatically registered because the connection can only be made by certified users. This registration is made by the registerUser method of the object *myDBCon* instantiated in the ServerProtocol class.

After receiving the Direct Trust file from the client, an object from the SPITEngineMain is instantiated to calculate the Reputation of each caller and to decide if it is a spammer or not. Then, this object retrieves a blacklist file with all spam callers determined by the Caller-REP Algorithm. This file is send back to the connected client. After the exchange of the information between the server and the client the connection is closed by the server.

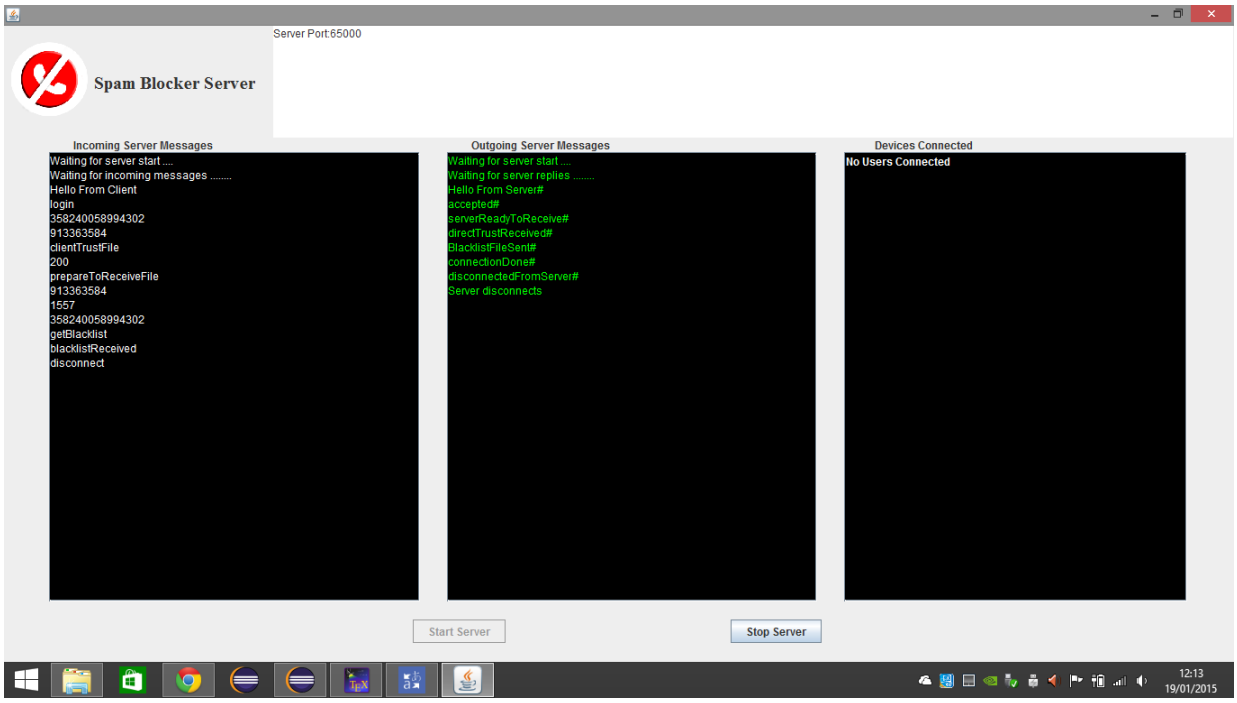

Figure 4.10: *Server Graphical Interface*

4.2 System Databases

4.2.1 Mobile Database

The Mobile database was developed taking in account the information needed for the overall system. Thus, its relational model is the following:

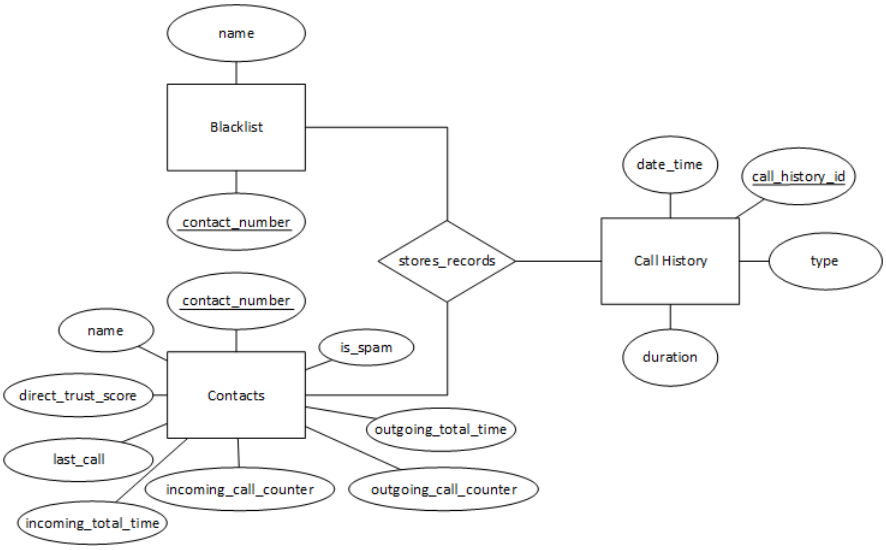

Figure 4.11: *Relational Model of the Mobile Application Database*

The Contacts table is the most important one, because is based on the columns incoming_total_time, incoming_calls_counter, outgoing_total_time and outgoing_calls_counter that the Direct Trust scores are calculated. Each time that a call is made or received, those values are updated. Thus, there is no need to process the Calls Log to compute the Direct Trust each time a call is perceived, avoiding long time processing to do such task.

The Call History table is where the call records are stored. This records are used to list the incoming, outgoing or blocked calls in the HistoryMain activity.

4.2.2 Server Database

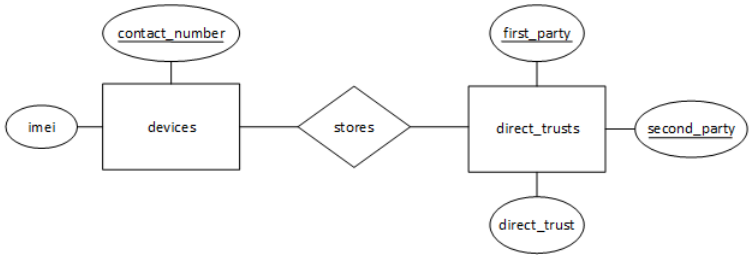

Figure 4.12: *Relational Model of the Server Database*

On the server side, the database consists only in two tables (see Fig. 4.12). The purpose of this database is to store the Direct Trust score of each number received from the users. Then, this values are used to create the server Direct Trust file which is used to compute the Global Reputation for every number stored in the database. Since each number can have different Direct Trust scores among the system users, the Direct Trust score can be easily updated if the unique record first_party-second_party exists, otherwise it is created a new entry on the direct_trusts table.

4.3 Use Cases

4.3.1 Show Contacts

1. The user is in the Main Screen and clicks the button *Contacts* (Fig. 4.13);
2. The contacts are listed showing the name, number, Direct Trust (if any) and last call date and time (Fig. 4.14).

Figure 4.13: *Main screen*

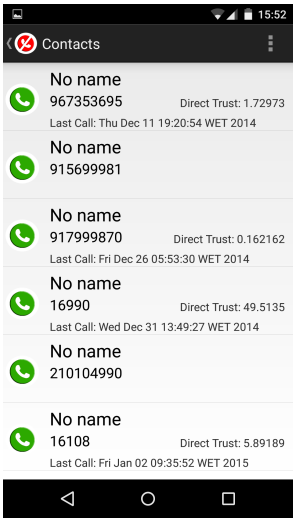

Figure 4.14: *Contacts screen*

4.3.2 Show Blacklist

1. The user is in the Main Screen and clicks the button *Blacklist* (Fig. 4.15);
2. The contacts are listed showing the name and number (Fig. 4.16);
3. If the user wants, he can:
   (a) Add a number manually: **Includes** Add number to blacklist icon that when pressed pop-ups a dialog (Fig. 4.17);
   (b) Long press an item to remove a number from the Blacklist: **Includes** Remove number from blacklist dialog (Fig. 4.19);

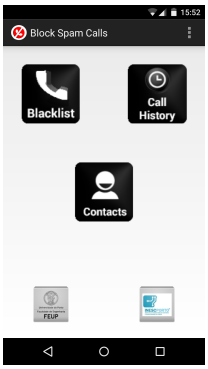

Figure 4.15: *Main screen*

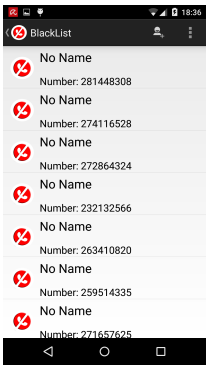

Figure 4.16: *Blacklist screen*

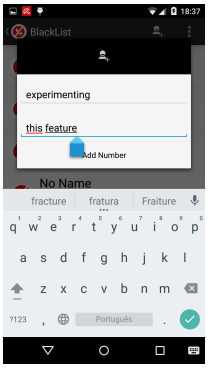

Figure 4.17: *Add contact to Blacklist dialog*

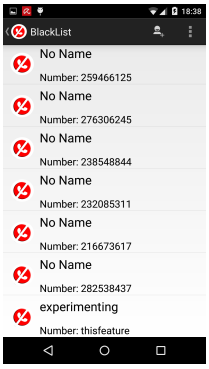

Figure 4.18: *Blacklist with new entry*

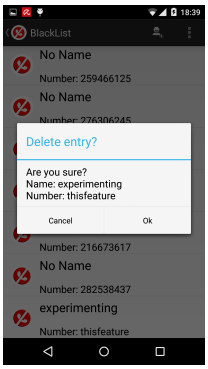

Figure 4.19: *Remove contact from Blacklist dialog*

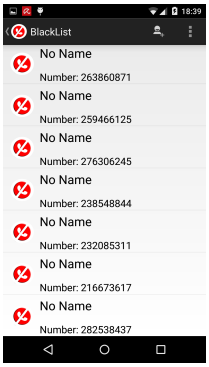

Figure 4.20: *Blacklist after removing*

4.3.3 Show History

1. The user is in the Main Screen and clicks the button *History* (Fig. 4.21);
2. The calls history is listed showing the name, number, last call date and time and the Direct Trust score (Fig. 4.22);
3. If the user wants, he can:
   (a) Click on a list entry: **Includes** Show the number history details (Fig. 4.23);

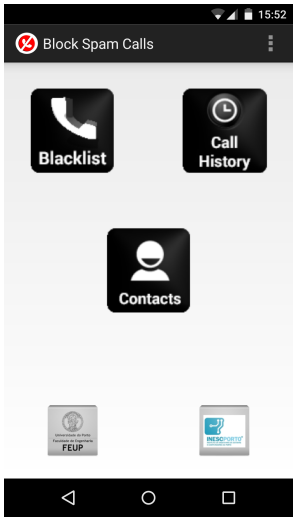

Figure 4.21: *Main Screen*

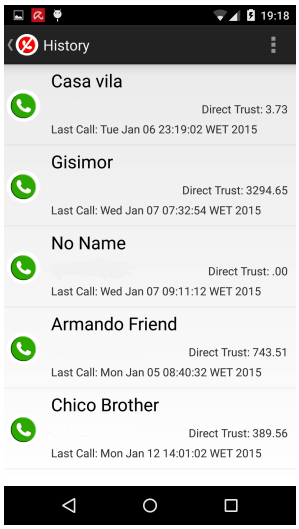

Figure 4.22: *History screen*

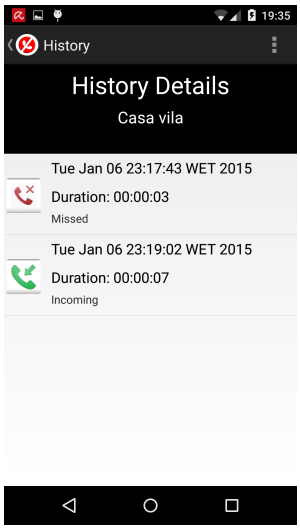

Figure 4.23: *History details*

4.3.4 Choose Settings

1. The user is in the Main Screen and clicks option *Settings* on the top right corner (Fig. 4.24);
2. If the user wants, he can:
   (a) Enable spam detection: **Includes** on/off switch (Fig. 4.25);
   (b) Connect to the server:
      i. **Includes** on/off switch (Fig. 4.26)
      ii. **Includes** Select synchronization interval dialog (Fig. 4.27);

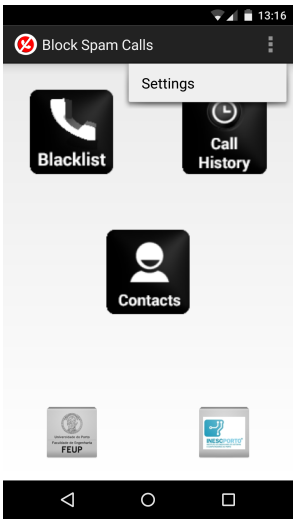

Figure 4.24: *Main Screen settings option*

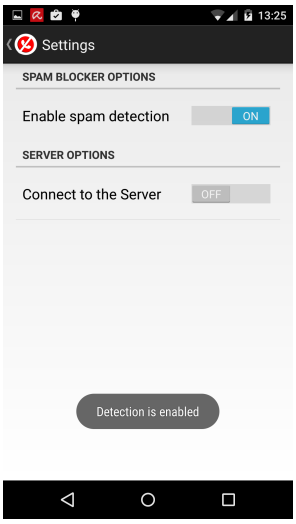

Figure 4.25: *Spam detection enabled*

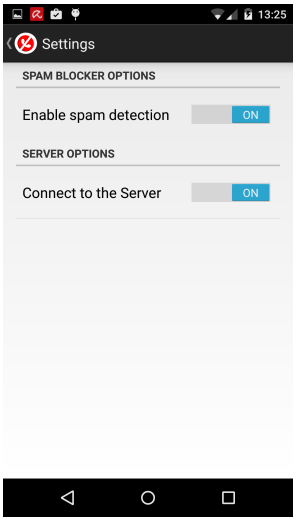

Figure 4.26: *Connect to the Server enabled*

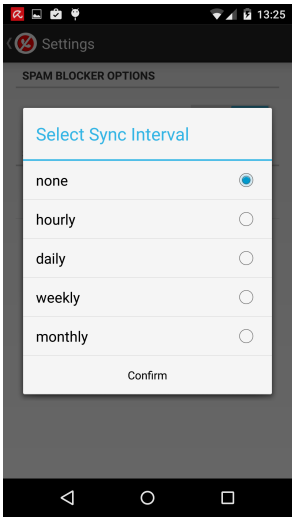

Figure 4.27: *Select synchronization interval*

4.3.5 Notification

1. The user receives a notification when the blacklist is received (Fig. 4.28);
2. If the user wants, he can:
   (a) Consult the blacklist: **Includes** selectable notification;
   (b) Drop the notification: **Includes** Android OS dismiss option;

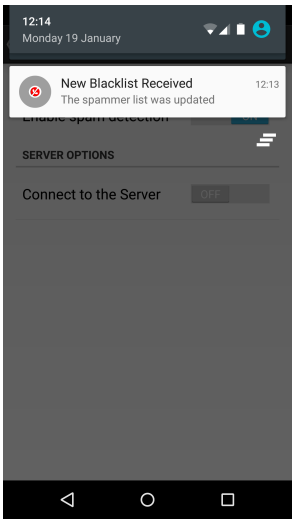

Figure 4.28: *Notification showed when the Blacklist is received*

For a better understanding about the connection with the server, the next figure presents the sequence diagram of that connection:

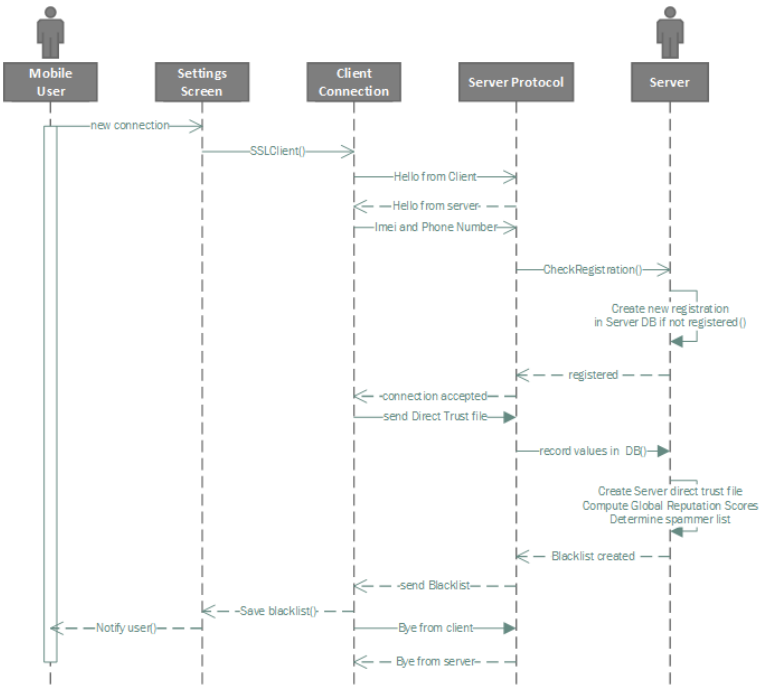

Figure 4.29: *Connection to the server sequence diagram*

4.4 Performance Results

There were made some performance tests to see which impact would the App have in the detection of spam calls while getting an incoming call and how long it would take to connect to the server and get the blacklist update.

The first results that will be shown are related to the time taken by the App to check if an incoming call is or is not a spammer and the call establishment delay. Those results can be observed in the following table:

Blacklist Size	20	100	500	1000	10000
Searching time (in ms)	0	0	1	9	70
Call establishment delay time (in ms)	9	9	21	41	110

Table 4.1: *Blacklist searching time*

It was observed that even if the size of the spammers list (Blacklist) was bigger than 1000 entries, the time to search that list was less than one second. When the detection call is enabled by the user, the blacklist records are copied from the database to an hash table. The values presented in the previous table were obtained using this method. On the other hand, when it was used a search directly from the mobile database the searching time did not diverge severely, for instance, for a blacklist with 1000 records the search time was in the order of 0.011 seconds.

The next results are related to the connection between the client and the Server. To obtain these time measures, there were made some random records in the Server database in order to simulate different amounts of clients registered in the system. The times achieved are represented in the Tab. 4.2.

Number of users	20	100	500	1000	10000
Client connection (seconds)	3.941	5.578	5.932	6.554	18.025
Blacklist Calculation (seconds)	0.064	0.405	0.643	1.294	11.894

Table 4.2: *Server time to compute the Global Scores and retrieve the Blacklist*

In the previous table, the Client connection refers to the time that it takes since its binding with the server until it stops. Thus, the amount of time referred in the table takes into account the sending of the Direct Trust file to the server, the storage of the records in the server database, the Blacklist calculation and the Blacklist retrieval to the client.

The Blacklist Calculation values take in account the running of the SPIT Engine, which calculates the Global-Rep of each number and decide which numbers are spammers.

It can be observed that above the 1000 users there was an increase of the client time connection and the Blacklist calculation. The possible cause of this is the weak processing power of the machine were the server is running (See Appendix A).

The bandwidth needed by the *App* to send the *Direct Trust* file and the one needed by the Server to retrieve the Blacklist was also measured. The Table 4.3 shows the results:

Size of the file (in lines)	20	100	500	1000	10000
Direct Trust (in KB)	0.527	3.123	16.188	42.220	303.133
Blacklist (in KB)	0.200	1.200	6.200	16.200	116.200

Table 4.3: *Bandwidth used when exchanging information with the Server*

We observed a small Bandwidth usage which means that the information exchanged doesn't affect the overall system performance, considering a maximum of 10000 lines for each file.

Finally, we checked how many memory did the *App* consume when the spam detection was enabled and how much storage amount did it occupy when installed. This information can be viewed in the Fig. 4.30 and 4.31, which were taken directly from the Android OS system settings in the Apps option.

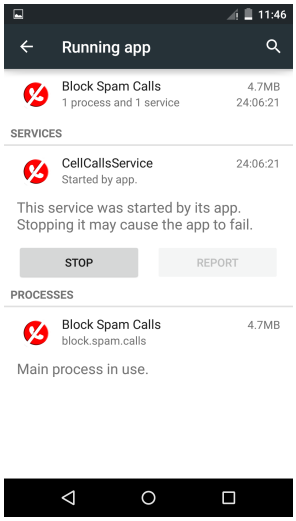

Figure 4.30: *Memory used*

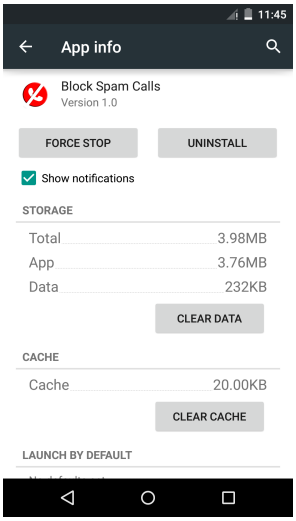

Figure 4.31: *Storage size*

The values observed were satisfactory and we realized that the *App* is really light in therms of system memory usage.

In the Battery Settings option we didn't find any reference to our *App* battery consumption which can mean one of two things: the *App* doesn't have a significant battery consumption when enabled or its battery consumption is not displayed.

Chapter 5

Conclusions and Framework Proposal

During the development of this *App* and the overall system implementation there were few issues related to to the implementation of some features. The constraints referred in the section 3.2 were some of them which does not allow the App to detect VoIP calls and calls from others applications such as Skype, Viber, etc. There was hurdle for getting call logs from these application because of non-availability of common api. So in this Application only makes the detection of cellular calls and the system was implemented taking in account the information gathered from those calls.

Another problem found was the way how the mobile phone should communicate with the server and the implementation of that connection. At first, it was hard to make the connection work due to the socket implementation and the way how it should be done. After fixing this problem, the difficulty found was the use of certificates to make a secure connection between the mobile phone and the server. This was due to the different types of certificates used by the mobile phone and the server.

Despite the problems previously appointed, this dissertation was very helpful to understand the Android OS API and to learn the Android API.

5.1 Accomplished goals

One of the goals achieved was the use of Caller-REP algorithm to determine if a certain number is or is not a spammer using direct feedback from the user without involving service providers. This algorithm allows the spam detection to be independent from the user and the operator.

The goal of making the blacklist using the Direct Trust scores of each callee was accomplished. To do so, we developed a suited protocol to exchange those values with the Server and then it computes the blacklist, retrieving it back to the user using a SSL connection.

The mobile phone uses that blacklist to block the incoming cellular calls, if the callee is in it.

One of the most important aspects of the App is the way that the Direct trust values are calculated, which are as follows:

1. When the *App* is first installed and used, the Calls history is processed to attribute the Direct Trust to each callee that contacted the user;

2. When the spam detection is enabled in the Settings screen, the Direct Trust is updated when a call is perceived.

It is based on the Direct Trust values that the blacklist is determined. Thus, there was extra care while programming this part of the code to assure that this value was accurately computed according to the system specifications.

Due to some improvements of the Android API, it were used some features only available for newer versions of the Android OS. Thus, the App is compatible with versions higher than 4.2.

5.2 Framework proposal

The work done in this dissertation can be used as groundwork for other voice communication mobile application (Skype, Viber) which can make use of some components developed, such as:

5.2.1 Direct Trust Module

Use the Direct Trust module to compute the Direct Trust score of each user contact. These values can be shared among the users and must be sent to the server in order to make the decision of which contact is spam or not.

It must be taken in account that each application can have different ways to identify their users, so this module should be modified to handle different types of contact identity, in order to be compatible with any application.

5.2.2 Server Module

This module should be used to store the Direct Trust Score sent by the users and to retrieve the blacklist based on those values.

Thus, the server must be located in a way that every application can access to it and get the blacklist.

It can be the Server that differentiates the several types of contacts ids, making to each application its own database and processing the Direct Trust scores accordingly to each application specifications.

5.2.3 Share Module

With this module, it's intended to have a feature of sharing the blacklist among the users who choose to have the spam detection enabled in their communication applications.

Thus, with this module the user can make his own blacklist and share it with the friends registered in the spam detection system. This feature permits the user to share his blacklisted numbers within his social network making this process more collaborative and those numbers can even be used to form the server blacklist.

Appendix A

Appendix

A.1 Laptop Specifications

- Processor: Intel® Core™ i5 3337U @ 1.80 GHz
- Operating System: Windows 8.1
- Memory: DDR3L 1600 MHz SDRAM, 4096 MB
- Storage: 2.5" SATA3 500GB HDD 5400
- Wi-Fi Card: Qualcomm Atheros AR9485 Wireless Network Adapter 802.11n

A.2 Mobile Phone Specifications

- Network: GSM / CDMA / HSPA / LTE
- Display: True HD IPS+ capacitive touchscreen; 16M colors; 4.95 inches; 1080 x 1920 pixels
- OS: Android OS, v5.0 (Lolipop)
- Chipset: Qualcomm MSM8974 Snapdragon 800
- Processor: Quad-core 2.3 GHz Krait 400
- Wi-Fi Card: 802.11 a/b/g/n/ac

References

[1] http://www.cfca.org.

[2] http://developer.android.com/index.html.

[3] http://grepcode.com/file/repository.grepcode.com/java/ext/com.google.android/android/1.5_r4/com/android/internal/telephony/ITelephony.java.

[4] http://stackoverflow.com/.

[5] Schmidt A, Leicher A, Shh Y, Cha I, and Guccione L. Sender scorecards. *IEEE Vehicular Technology Magazine*, pages 52 – 59, March 2011.

[6] M.A. Azad and R. Morla. Caller-rep: detecting unwanted calls with caller social strength. *Computers & Security*, pages 219 – 236, November 2013.

[7] Muhammad Ajmal Azad and Ricardo Morla. Multistage spit detection in transit voip. *Software, Telecommunications and Computer Networks (SoftCOM), 2011 19th International Conference on*, pages 1–9, 2011.

[8] Muhammad Ajmal Azad and Ricardo Morla. Privacy-aware collaborative spit detection system. *Submitted to IEEE Transaction on Secure and Dependable Computing*, 2011.

[9] Shin D, Ahn J, and Shim C. Progressive multi gray-leveling: a voice spam protection algorithm. *IEEE Network*, pages 18 – 24, 2006.

[10] Himanshu Dwivedi. *Hacking VoIP*. No Starch Press, Inc., 555 De Haro Street, Suite 250, San Francisco, CA 94107, USA, 2009.

[11] Bokharaei H, Sahraei A, Ganjali Y, Keralapura R, and Nucci A. You can spit, but you can't hide: spammer identification in telephony networks. *2011 IEEE INFOCOM*, pages 41 – 45, April 2011.

[12] Sengar H, Wang X, and Nichols A. Thwarting spam over internet telephony (spit) attacks on voip networks. *Quality of service (IWQoS), 2011 IEEE 19th international workshop*, pages 1 – 3, June 2011.

[13] Sengar H, Wang X, and Nichols A. Call behavioral analysis to thwart spit attacks on voip networks. *Security and privacy in communication networks, serLecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 501 – 510, 2012.

[14] Ono K and Schulzrinne H. Have i met you before?: using cross-media relations to reduce spit. *3rd International conference on principles, systems and applications of IP telecommunications*, pages 1–7, September 2009.

[15] Hansen M, Hansen M, Mller J, Rohwer T, Tolkmit C, and Waack H. Developing a legally compliant reachability management system as a countermeasure against spit. *Third annual VoIP security workshop*, 2006.

[16] Kolan P and Dantu R. Socio-technical defense against voice spamming. *ACM Transactions on Autonomous and Adaptive Systems*, March 2007.

[17] Dantu R and Kolan P. Detecting spam in voip networks. *The steps to reducing unwanted traffic on the internet - USENIX Association*, pages 31 – 37, July 2005.

[18] Jonathan Rosenberg and Cullen Jennings. The Session Initiation Protocol (SIP) and Spam. RFC 5039, IETF, January 2008.

[19] Jonathan Rosenberg, Henning Schulzrinne, Gonzalo Camarillo, Alan Johnston, Jon Peterson, Robert Sparks, Mark Handley, and Eve Schooler. SIP: Session Initiation Protocol. RFC 3261, IETF, June 2002.

[20] Balasubramaniyan V, Ahamad M, and Park H. Callrank: combating spit using call duration, social networks and global reputation. *Fourth conference on email and anti-spam (CEAS2007)*, pages 501 – 510, August 2007.

[21] Ted Wallingford. *Switching to VoIP*. O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472, USA, 2005.

[22] Rebahi Y, Sisalem D, and Magedanz T. Sip spam detection. *International conference on digital telecommunications*, pages 68 – 74, August 2006.

[23] Wu Y-S, Bagchi S, Singh N, and Wita R. Spam detection in voice-over-ip calls through semi-supervised clustering. *39th annual IEEE/IFIP international conference on dependable systems and networks (DSN)*, pages 307 – 316, June 2009.